PROPHET MODEL – DOCUMENTATION

Prophet model:

The Prophet model, developed by Facebook's AI Research team, is a robust statistical time series forecasting tool. It handles complex time series data characterized by trends, seasonality, and holiday effects.

Key Features:

- Trend Modeling: Prophet can model linear and non-linear trends, adapting to changes in the underlying data.
- Seasonality Detection: It automatically detects and models yearly and weekly seasonality and custom seasonality patterns.
- Holiday Effects: Prophet allows you to incorporate the impact of holidays on your time series data, which can significantly improve forecast accuracy.
- Changepoint Detection: Prophet can automatically detect and model changes in trend and seasonality over time.

How it Works:

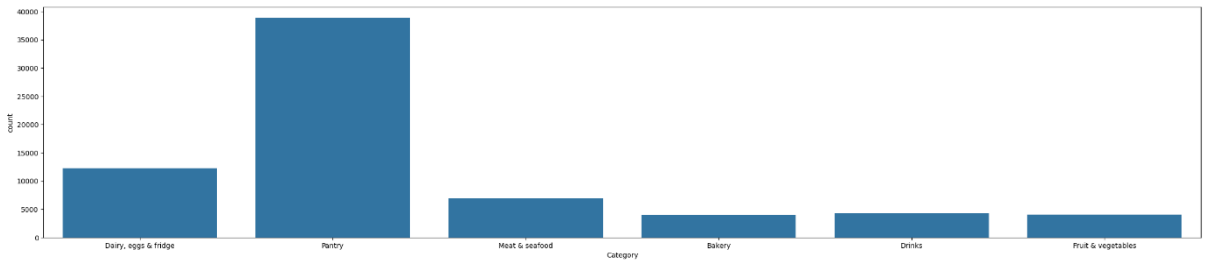
- Data Preparation: The model requires historical time series data with a date column and a value column.
- Model Training: The model is trained on the historical data to learn the underlying patterns and trends.
- Future Forecasting: Once trained, the model can generate forecasts for future periods, taking into account the detected trends, seasonality, and holiday effects.
- Uncertainty Quantification: Prophet provides uncertainty intervals around the forecasts, giving you a sense of the potential variability in future outcomes.

By effectively capturing these key components, Prophet provides accurate and reliable forecasts, making it a valuable tool for businesses and organizations that need to plan.

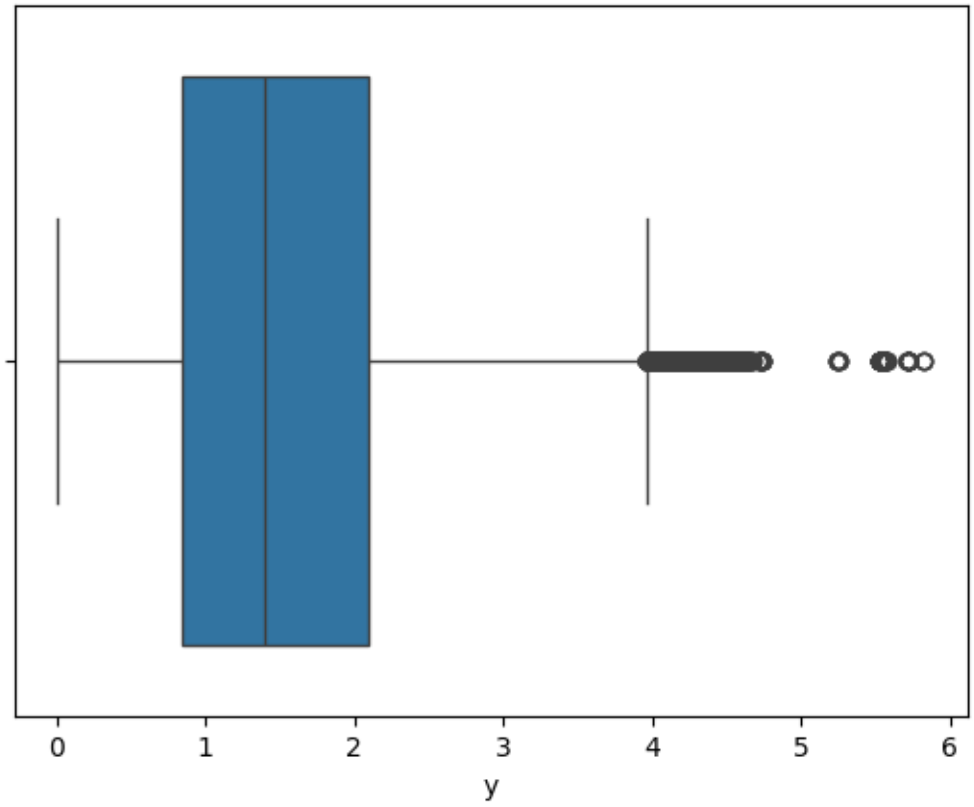
Code explanation:

- Imports:

    - *pandas as pd*: Imports the pandas library for data manipulation.
    - *from prophet import Prophet*: Imports the Prophet library for time series forecasting.
    - *matplotlib.pyplot as plt*: Imports the matplotlib library for creating plots.
    - *seaborn as sns*: Imports the seaborn library for creating statistical visualizations.
    - *numpy as np*: Imports the numpy library for numerical operations.
    - *from sklearn.metrics import mean_squared_error, mean_absolute_error*: Imports the mean_squared_error and mean_absolute_error functions from the scikit-learn library for evaluating model performance.
- Data Loading and Preprocessing:
    - Load the data:
        - *data = pd.read_csv("/content/Aus_grocery_synthetic_dataset2.csv")*: Reads the grocery dataset from a CSV file.
        - *df = pd.DataFrame(data):* Converts the loaded data into a pandas DataFrame.
    - Format the date and target variable:
        - *df['ds'] = pd.to_datetime(df['RunDate'], errors='coerce')*: Converts the 'RunDate' column to a datetime format compatible with Prophet.
        - *df['y'] = df['unit_price_x']*: Selects the 'unit_price_x' column as the target variable for forecasting.
- Sort by date:
  df = df.sort_values(by='RunDate'): Sorts the DataFrame by the 'RunDate' for chronological order.
- Visualize category distribution:
  This code block creates a bar chart to show the frequency of each category in the dataset using seaborn.

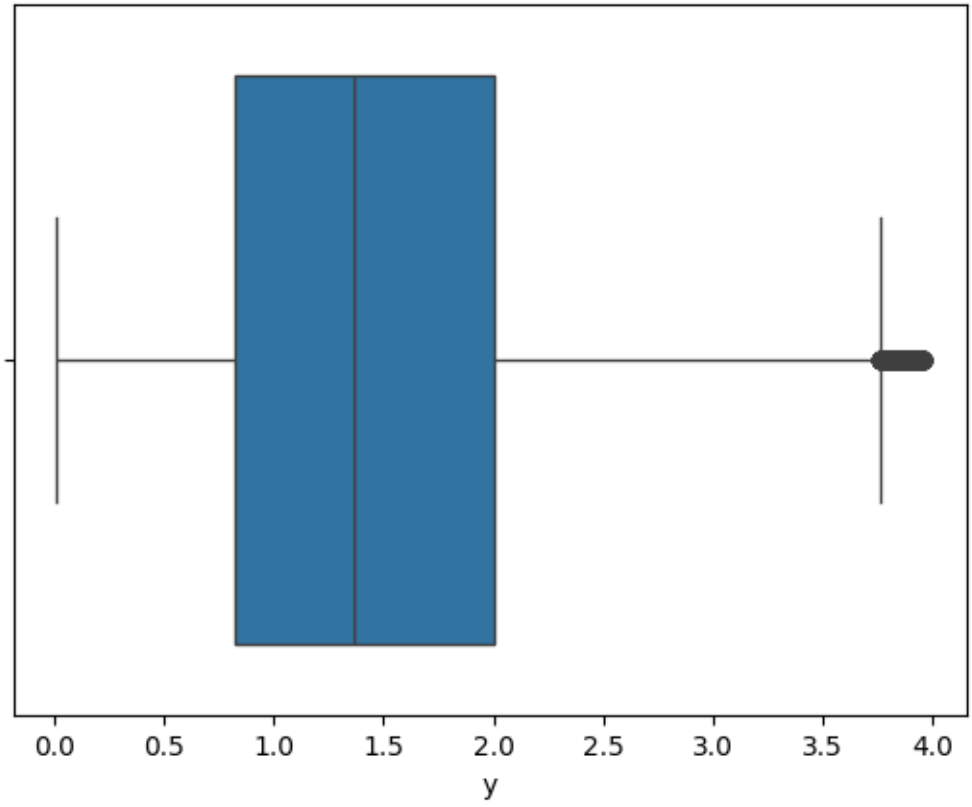


- Apply log transformation:
  df['y'] = np.log1p(df['y']): Applies a logarithmic transformation (log1p) to the target variable. This can help normalize skewed data.
- Detect outliers:
  This code block uses boxplots to identify outliers in the target variable.



- Remove outliers:
  - *Q1 = df['y'].quantile(0.25)*: Calculates the first quartile (Q1) of the target variable.
  - *Q3 = df['y'].quantile(0.75)*: Calculates the third quartile (Q3) of the target variable.
  - *IQR = Q3 - Q1*: Calculates the interquartile range (IQR).
  - *lower_bound = Q1 - 1.5 * IQR*: Defines the lower bound for outlier detection.

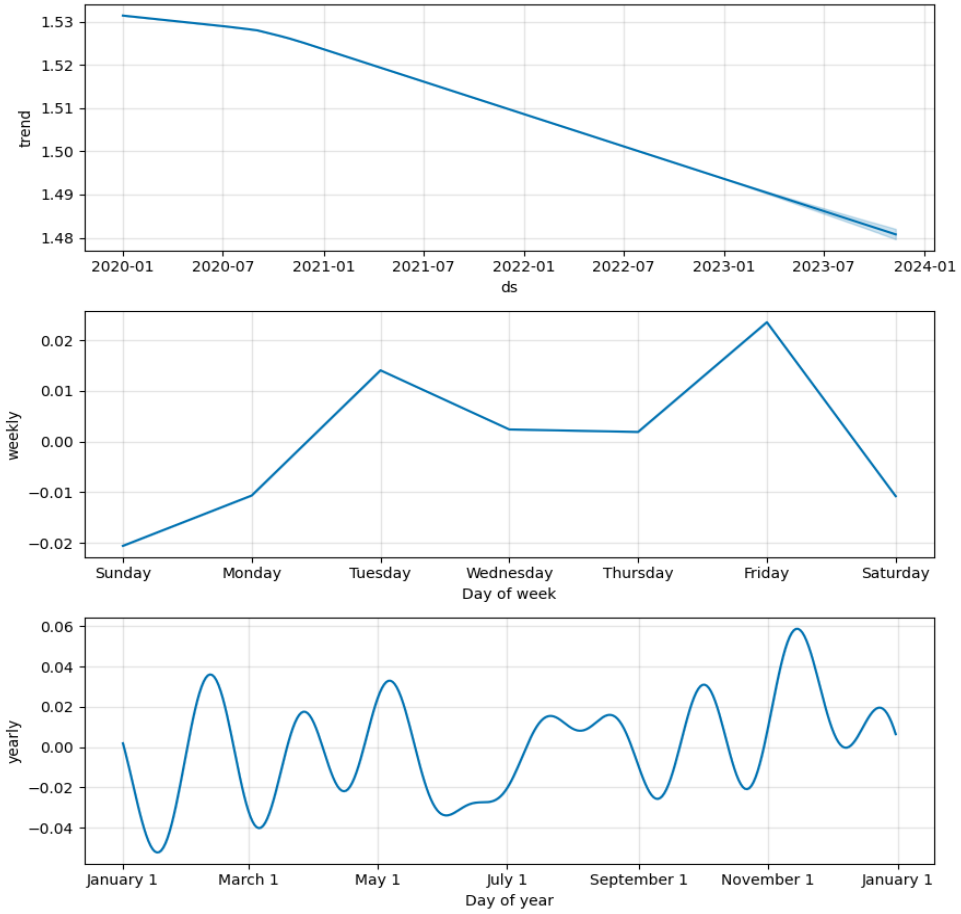
- *upper_bound = Q3 + 1.5 * IQR*: Defines the upper bound for outlier detection.
- *df = df[(df['y'] >= lower_bound) & (df['y'] <= upper_bound)]*: Filters the DataFrame to remove outliers beyond the defined bounds.



- Handle missing values:
  df = df.dropna(subset=['y', 'ds']): Removes rows with missing values in the 'y' (target variable) and 'ds' (date) columns.
- Select data for Prophet model:
  df_prophet = df[['ds', 'y']]: Creates a new DataFrame df_prophet containing only the 'ds' (date) and 'y' (target variable) columns required by the Prophet model.
- Splitting Data into Train and Test Sets:
  - *train = df_prophet.iloc[:-365]*: Splits the data into a training set containing all rows except the last 365 (one year of data).
  - *test = df_prophet.iloc[-365:]*: Splits the data into a test set containing the last 365 rows (one year of data) for evaluating the model's performance.
- Building and Fitting the Prophet Model:

    - Create the Prophet model:
      model = Prophet(): Initializes a Prophet model object.
    - Fit the model on the training data:
      model.fit(train): Trains the Prophet model using the training dataset.
- Generating Future Dates and Predictions:
    - Create a DataFrame for future dates:
      future = model.make_future_dataframe(periods=365): Creates a DataFrame containing future dates for the next 365 days (one year) based on the historical data.
    - Make predictions:
      The trained model is used to make predictions using the *predict ()* method.
- Understanding the forecast DataFrame:
  The forecast DataFrame contains the following columns:
    - *ds:* The datetime index.
    - *yhat:* The predicted value.
    - *yhat_lower:* The lower confidence interval for the prediction.
    - *yhat_upper:* The upper confidence interval for the prediction.
    - *trend:* The trend component of the forecast.
    - *seasonal:* The seasonal component of the forecast.
    - *holiday_weights:* The weights of the holidays on the forecast.

- Visualizing the Forecast:



  - Weekly Trend (Day of the Week):
    Key Insight: Discounts or price variations appear to be influenced by the day of the week.
    Tuesday: The weekly plot shows a spike on Tuesday, suggesting higher prices or reduced discounts.
    Friday: Another spike occurs, indicating possibly higher demand and reduced discounts before the weekend.
    Sunday and Monday: These days have the lowest values, indicating the possibility of higher discounts or lower prices to attract customers at the start of the week.
  - Yearly Trend (Seasonal Variations):
    Key Insight: The yearly trend reveals cyclic patterns in prices across different months.
    Peaks: Noticeable price increases occur around March, July, and November-December.
    Dips: Lower prices are observed in January, May, and September.

    - Trend Over Time (General Price Direction):
      Key Insight: The overall trend shows prices rising until mid-2020, then declining steadily afterward. This could indicate that the supermarket:
        - Initially increased prices (possibly due to inflation or higher costs).
        - Later offered more discounts or price cuts due to competitive pressures or reduced demand.
- Evaluating the Model:
  To evaluate the performance of the model on the test set, I calculated metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE):
  MSE: 0.86, RMSE: 0.93, MAE: 0.74
  The error metrics are low, indicating the model is performing reasonably well on this dataset. RMSE is slightly higher than MAE, suggesting there might be some predictions with larger errors (outliers).